IBM® Technical Disclosure Bulletin Vol. 38 No. 12 December 1995

Non-V latile Cach Storage Allocation Algorithm

Disclosed is a cache allocation algorithm which removes present Non-Volatile Storage size constraints from cache storage control subsystems (e.g., IBM* 3990) and so allows greater performance than is presently attainable. This algorithm allows the modified data area of the non-volatile cache to expand and contract in response to fluctuations in the frequency of DASD Fast Write channel programs, constrained only by the total cache size.

Cache Nodes - The configuration consists of two or more ***cache nodes*** Each cache node has its own microprocessor, battery backed-up (non-volatile) cache and power supply. In the case of a power outage at the storage control, the modified data from all cache nodes is dumped to DASD.

Primary and Journal Cache Storage - Cache Storage at any cache node can be used as storage for both track images and for modified records. Track image storage is named Primary storage, and modified record storage is named Journal storage. There is no fixed partitions between Primary and Journal storage, as will be discussed in the ***Allocation*** section.

The storage control attempts to distribute the amount of data allocated over the several cache nodes, in a balanced manner. When the subsystem powers up, the SEASTAR control code chooses a Primary and Journal cache node for each CKD track in the subsystem, ensuring that Primary and Journal cache data will be allocated to separate ***failure boundaries.***

Write data, e.g., an Update Write, modifies a record or records in a track image, and is written to Journal storage in a separate cache node. Thus, there are two copies of the modified data, and each copy resides in a separate cache node, on a separate failure boundary.

Cache Allocation

- A cache node has its cache partitioned into different size segments. Segments of the same size are chained together to form a pool of segments of a given size. E.g., segment sizes could be 256 bytes, 1 KB, 2KB, 4KB, 8KB, etc., up to a maximum number of pools.
- Record fields are allocated to available best fitting segment. Record fields that are stored in segments are the Key field (if it exists), and the Data field (the Count field, which is a fixed size, is stored in the record's ***Track Descriptor Vector*** element, which is described in a subsequent paragraph). There are no contiguity restrictions on the allocation of the fields for an individual record. ***Not more than one field can be stored in a cache segment***, although a field can consume multiple segments.

  This method of cache allocation gives great flexibility at the cost of possibly some wasted space in individual segments.
- When a cache miss occurs, the following happens:

Dec. 1995 ... IBM TECHNICAL DISCLOSURE BULLETIN ... <-- pp39-42 -->

Title: # Non-Volatile Cache Storage Allocation Algorithm

Text

Disclosed is a cache allocation algorithm which removes present Non-Volatile Storage size constraints from cache storage control subsystems (e.g., IBM* 3990) and so allows greater performance than is presently attainable. This algorithm allows the modified data area of the non-volatile cache to expand and contract in response to fluctuations in the frequency of DASD Fast Write channel programs, constrained only by the total cache size.

Cache Nodes - The configuration consists of two or more cache nodes Each cache node has its own microprocessor, battery backed-up (non-volatile) cache and power supply. In the case of a power outage at the storage control, the modified data from all cache nodes is dumped to DASD.

Primary and Journal Cache Storage - Cache Storage at any cache node can be used as storage for both track images and for modified records. Track image storage is named Primary storage, and modified record storage is named Journal storage. There is no fixed partitions between Primary and Journal storage, as will be discussed in the Allocation section.

The storage control attempts to distribute the amount of data allocated over the several cache nodes, in a balanced manner. When the subsystem powers up, the SEASTAR control code chooses a Primary and Journal cache node for each CKD track in the subsystem, ensuring that Primary and Journal cache data will be allocated to separate failure boundaries.

Write data, e.g., an Update Write, modifies a record or records in a track image, and is written to Journal storage in a separate cache node. Thus, there are two copies of the modified data, and each copy resides in a separate cache node, on a separate failure boundary.

Cache Allocation

o A cache node has its cache partitioned into different size segments. Segments of the same size are chained together to form a pool of segments of a given size. E.g., segment sizes could be 256 bytes, 1 KB, 2KB, 4KB, 8KB, etc., up to a maximum number o pools.

o Record fields are allocated to available best fitting segme Record fields that are stored in segments are the Key field (i it exists), and the Data field (the Count field, which is a fixed size, is stored in the record's Track Descriptor Vector element, which is described in a subsequent paragraph). There are no contiguity restrictions on the allocation of the fields for an individual record. Not more than one field can be stored in a cache segment, although a field can consume multiple segments.

This method of cache allocation gives great flexibility at the cost of possibly some wasted space in individual segments.
o When a cache miss occurs, the following happens:
1. If the accessed track image does not exist in Cache Storag a Track Descriptor Vector (TDV) is allocated. The TDV resembles the 3990 Track Information Block. For every record
stored in the track image it has a chain of pointers emanating from the record's TDV element to the so-called Buffer Control Blocks (BCBs). The BCB's in turn contain descriptive data (e.g., the field stored), a pointer to the segment containing all or part of the field, and a pointer to
the next BCB, if any, which could contain more of the same field or the next field of the record.
2. The TDVs can be sparse (contain few or discontiguous recor control information), can have their contents modified over time due to record read misses, due to record write misses, or due to partial track front end read misses.
3. For a track with modified records, the TDV for the Primary storage allocation in one cache node points to its associated TDV for the Journal storage allocation in a second
cache node.
4. After storage segments are allocated in the one or two cac nodes, the data is either staged to the segments (read miss)
or written to the segments from the channel.
Note: Data is not normally staged from DASD to cache on a write miss. Write miss data is written directly to Primary and Journal cache, after the record format is checked against
a table of track formats, called Track Format Descriptor table. Only in the exceptional case when a valid Track Format Descriptor is not found is the target track staged from DASD.
Modified Data
o As already mentioned, modified records reside in Primary an Journal Cache Storage, and these two areas must be allocated t separate cache nodes.
o Update in place never overlays the home record. Rather, ne TDVs
and segments are allocated in both the Primary and Journal areas,
the fields are written, and then the TDVs are merged with existing TDVs, where the former contain additional records tha were not the target of the update.
o The obsolete (earlier version) record has its space dealloc
o This method of update and allocation eliminates the problem track image lockout experienced by the 3990 whenever there is an
attempt to read or write a track image during a destage of modified data. Therefore,
1. A read is allowed to execute to a record being destaged, since its cache track image contents are not being disturbed
(i.e., neither the channel read nor the lower interface read
disturbs the data contents).
2. A write is always to a new cache segment allocation,

therefore, the old record data can be destaged to DASD while
the updated version is written to a newly allocated segment.
Of course, in this case the just destaged version of the record is now obsolete.
Least Recently Used, Demotion and Destage
o Each cache node has its own Least-Recently-Used (LRU) list composed of track entries. Journal data are also controlled b the LRU list through indirection: through the Primary TDVs pointing to the associated Journal TDVs.
o In a pure LRU discipline, destaging of modified data is triggered
by the track approaching the bottom of the LRU list. When the track reaches the bottom of the LRU list, after destage of modified data has occurred, the track is demoted from the Cach Storage.
Modified records to destage are selected with reference to their position on the LRU list. However, the number of modified
records to destage is determined by consideration of space requirements dictated by other objectives, which are listed below. These objectives constrain the operation of a pure LRU discipline with respect to its influence on the amount of modified data maintained in the cache. The objectives constraining the LRU influence are:
1. Maintain reasonable read hit ratios.
2. Maintain space so that in the case of a node failure Prima and Journal data from the failed cache node can be copied to
operational cache nodes.
3. Limit the magnitude of Journal data so that it does not exceed the DASD dump space limits.
4. Meet RAID 5 and Log-Structured Array space requirements, which impose additional restrictions on the amount of modified data stored in cache:
- Cache space is required for parity data.
- Cache space is required for copies of old data in the execution of the RAID 5 algorithm.
- Additional data must be staged because of DASD SCSI fixed block formatting requirements.
o In addition, TDV record modifications are time stamped, and these
provide a handle for destaging modified records before they ag beyond a given threshold since first update, and also allow th freeing of Journal space for modified tracks that continue to be
read (i.e., tracks that stay near the top of the LRU list due only to frequent reads).
Highlights of the Non-Volatile Cache Storage Allocation Algori
o Multiple cache nodes, each with non-volatile Cache Storage individual processors.
o Allocation of a track's modified data to a separate node th that containing the track image data: this is the separation of
Journal storage allocations from Primary allocations. Althoug both types of allocations exist in the same cache node, for a particular track these data must be on separate cache nodes. Nevertheless, there continues to be two copies of the modified data, one copy in each cache storage type.

o Variable size cache segments with not more than one record allocated to a segment. This is a great advance in the flexibility of cache storage segmentation and makes the Generalized Record Caching algorithm, for one, much more feasible
than on the 3990.
o LRU list controlled magnitude of Cache Storage allocated to modified data, subject to recovery and read hit ratio constraints. This is a realization of the main thrust of this publication.
o Elimination of lockout of read or write channel program access to
data during the destage of the modified data on a track.
* Trademark of IBM Corp.

Diagrams: None

Order/Fcode/Docket: **95A 62381 / / TU8930197**
Vol=38 PubNo=12

: **Dec. 1995** ... IBM TECHNICAL DISCLOSURE BULLETIN ...
<– pp39-42 –>

pg. 42